Hindawi
Security and Communication Networks
Volume 2022, Article ID 1249393, 16 pages
https://doi.org/10.1155/2022/1249393

Research Article

k Nearest Neighbor Similarity Join Algorithm on High-Dimensional Data Using Novel Partitioning Strategy

Youzhong Ma,[1,2] **Qiaozhi Hua**,[3] **Zheng Wen**,[4] **Ruiling Zhang**,[1] **Yongxin Zhang**,[1] **and Haipeng Li**[5]

[1]*School of Information and Technology, Luoyang Normal University, Luoyang 471934, China*
[2]*Henan Key Laboratory for Big Data Processing & Analytics of Electronic Commerce, Luoyang 471934, China*
[3]*Computer School, Hubei University of Arts and Science, Xiangyang 441000, China*
[4]*School of Fundamental Science and Engineering, Waseda University, Tokyo 169-8050, Japan*
[5]*Capinfo Company Ltd., Beijing 100010, China*

Correspondence should be addressed to Qiaozhi Hua; 11722@hbuas.edu.cn

Received 17 January 2022; Accepted 22 March 2022; Published 26 April 2022

Academic Editor: Thippa Reddy G

k nearest neighbor similarity join on high-dimensional data has broad applications in many fields; several key challenges still exist for this task such as "curse of dimensionality" and large scale of the dataset. A new dimensionality reduction scheme is proposed by using random projection technique, then we design two novel partition strategies, including equal width partition strategy and distance split tree-based partition strategy, and finally, we propose *k* nearest neighbor join algorithm on high-dimensional data based on the above partition strategies. We conduct comprehensive experiments to test the performance of the proposed approaches, and the experimental results show that the proposed methods have good effectiveness and performance.

1. Introduction

With the rapid development of emerging technologies such as big data [1, 2], Internet of things [3, 4], Deep Learning [5, 6], Adversarial Training [7, 8], Federated Learning [9], and 5G [10–12], the smart healthcare systems are becoming more and more pervasive and necessary in modern hospitals, and massive and diverse medical data have been accumulated by using a great volume of wearable sensors, the Internet of Medical Things [13, 14], or the Internet of Health Things [15, 16]. Medical data analysis, security, and privacy protection [17] are very important for using massive medical data. Many research works have been done, which can be used as the references to analyze the medical data and secure the Internet of Medical Things, such as the blockchain-based security approaches [18, 19], the security technologies in Internet of Internet of Vehicles [20, 21], Industrial Internet of Things [22–25], Artificial Intelligence of Things [26–28], Energy Internet [29], and Intelligent transportation [30, 31]. Similarity join operation plays an important role in medical data analysis [32, 33]. Threshold-based similarity join query on high-dimensional data can obtain all the data pairs whose distance is less than or equal to the given distance threshold, and it needs to know the distance threshold in advance; however, in many cases, it is hard or even impossible to get the distance threshold in advance, while *k* nearest neighbor similarity join (*k* NNJ) does not need to obtain the distance threshold in advance. *k* NNJ is always used as the preprocessing stage for classification or clustering task and has broad real applications in many data mining tasks, such as multimedia analysis, spatial data mining, time series, data streams, and social network. Taking similar medical image pairs detection as an example, in some cases, it is hard to make a definite diagnosis according to the medical images only; however, it is possible for us to obtain some similar medical images of the existed confirmed cases (*k* nearest neighbors) to help the doctors to make a final judgement on the unconfirmed diseases.

In many applications, the target objects can be represented as vector forms through feature extraction in order to facilitate data processing, such as time series, videos, and trajectories, especially for the image processing tasks [34, 35]. With the continuous improvement of the accuracy of data acquisition equipment, the dimensionality of vectors representing the target objects will be very high, maybe hundreds of dimensions or even tens of thousands of dimensions. The calculation of the similarity or distance between the object pairs is a time costly operation. Most of the existed approaches conduct *k* NN join operation directly on the original dimensional space, so their performance is not ideal. It is an effective way to reduce the time cost of *k* NN join through reducing the dimensionality of the original data points. There are three contributions in the paper:

(i) We proposed an effective dimensionality reduction approach that can make sure that the data points in the projected space preserve the location relationship to some extent as in the original space.

(ii) We proposed two partition strategies, including equal width partition strategy and distance split tree-based partition strategy, and a novel *k* nearest neighbor join algorithm was proposed by using the above two partition strategies.

(iii) Comprehensive experiments are conducted, and the final results prove that our approaches have better effectiveness and performance.

The other parts of the paper are arranged as follows. The detailed related research works are described in Section 2. Section 3 displays the notations, problem definitions, and some theorems. The lower bound probability is figured out in Section 4. The *k* nearest neighbor join algorithm using random projection and partition strategies is described in Section 5. The detailed experimental results of the proposed approaches are displayed in Section 6. Section 7 makes a conclusion of the paper and points out the future research directions.

2. Related Works

Many researchers have conducted in-depth research on similarity join problems because of their broad applications and important role in data mining or machine learning context; several survey articles have conducted a comprehensive and detailed analysis of the similarity join problem literature [36, 37].

k nearest neighbor join: an approximated *k* NN similarity join approach in metric spaces was proposed by Ferrada et al. [38], its time complexity is $\Theta(n^{3/2})$, and the empirical precision can reach up to 46%. Lu et al. [39] designed a novel approach called PCBJ by using Voronoi diagram, which can deal with exact *k* NN similarity join problems; however, its performance is not very good as the increase of data dimensionality. Dai et al. [40] proposed two novel *k* NN join algorithms based on the MapReduce framework, which are DSGMP-J using Distributed Sketched Grid and VDMP-J using Voronoi diagram; DSGMP-J [40] approach is easy to implement, but it ignores the real distribution of the dataset. Zhao et al. [41] designed an effective data partitioning scheme called *k* NN-DP, which can solve the load imbalance issues caused by the data skewness problem; two novel schemes called LSH+ and *z*-value+ were developed based on *k* NN-DP to deal with *k* NN join operations under MapReduce framework. Song et al. [42] conducted a detailed analysis of the common workflows of the several existing *k* NN algorithms and further analyzed their load balancing, accuracy rate, and overall complexity; finally, a choice guideline was given which can help select the suitable methods for a specific case. Song et al. [43] also conducted a detailed comparison among the existing *k* NN join approaches both theoretically and experimentally on the MapReduce platform. RankReduce approach [44] was proposed by using locality-sensitive hashing with MapReduce for processing *k* nearest neighbor query. Hu et al. [45] proposed an adaptive *vk* NN join algorithm by using the Voronoi diagram, which can eliminate many unnecessary computations. There still exist many other research works about *k* NN join problems in several applications, such as Trajectory Data [46], machine learning [47], and Hilbert R-tree-based *k* NN join algorithms [48].

Top-*k* similarity join: Kim et al. [49] proposed two serial algorithms called the "divide-and-conquer algorithm" and "branch-and-bound algorithm" using the MapReduce framework, which can deal with Top-*k* similarity join problems efficiently. Chen et al. [50] developed a new distance function based on LSH and proposed an RDD-based Top-*k* similarity join algorithm using the Spark framework, and the test results proved that the RDD-based algorithm has better scalability and effectiveness than that of Hadoop. Ma et al. [51] developed a Top-*k* threshold estimation approach through sampling and designed an effective filtering solution by using the Symbolic Aggregate Approximation technique and then proposed a SAX-Top-*k* algorithm, which can deal with Top-*k* similarity join problem. Lei et al. [52] explored the similarity join problems for massive probabilistic dataset. The main idea of Lei et al. [52] is mapping the probabilistic data from the original space to the reduced dimensional space (one dimension), and then the range query on the one-dimensional space can be instead of the threshold-based similarity join query on the original space. Based on the above schemes, the authors proposed the Top-*k* Block Nested Loop Join Algorithm and Top-*k* Data Locality Preserving Join Algorithm, respectively. MELODY-JOIN [53] can improve the efficiency of the Top-*k* join on the histogram probabilistic dataset by using the standard lower bound space of the EMD distance; however, it cannot deal with the data skew problem efficiently. EMD-MPJ [54] proposed a novel Mapping-based Data Partitioning Framework that can solve the data skew problem. Heads-Join [55] made an extension to MELODY-JOIN [53] so that it can deal with both range similarity join and Top-*k* similarity efficiently.

Threshold-based similarity join: Cristiani et al. [56] designed a novel randomized set join method whose recall can be up to 100%, and its performance is better than that of the existing approach theoretically and empirically. Gowanlock et al. [57] proposed several novel methods to accelerate the similarity self-join by making full use of the power and characteristic of GPU. There still exist some other research works which focus on the similarity join problem using GPU [58]. Sandes et al. [59] developed a novel filtering scheme that can speed up the exact set similarity join more efficiently. Ding et al. [60] exploited the privacy preserving problems in similarity join using MapReduce context. Wu et al. [61] proposed a novel parallel framework called SMS-Join which can support similarity join operations in metric space using the MapReduce paradigm. Ma et al. studied the similarity join problems for high-dimensional dataset through developing a novel dimension reduction approach based on the Piecewise Aggregate Approximation technique and proposed two algorithms called SAX-Based HDSJ [62] and Mp-V-SJQ [63].

3. Preliminaries

3.1. Notations. The notations used in this paper are listed in Table 1.

3.2. Problem Definition. The definitions of KNN and KNN join will be described in this subsection. Given two datasets $R \in \mathfrak{R}^d$ and $S \in \mathfrak{R}^d$, $\mathbf{R} = \{r_1, r_2, \ldots, r_{n_1}\}$, $\mathbf{S} = \{s_1, s_2, \ldots, s_{n_2}\}$, $|\mathbf{R}| = n_1$, and $|\mathbf{S}| = n_2$. r_i is the i_{th} data point from $\mathbf{R}$, $r_i = \langle r_{i1}, r_{i2}, \ldots, r_{i\,d}\rangle, s_j$ is the j_{th} data point from $\mathbf{S}$, $s_j = \langle s_{j1}, s_{j2}, \ldots, s_{j\,d}\rangle$, and the distance measurement used in the paper is the Euclidean distance denoted as $dist(r_i, s_j)$,

$$dist(r_i, s_j) = \sqrt{\sum_{l=1}^{d} (r_{il} - s_{jl})^2}, \quad (1)$$

where $dist(r_i, s_j) > \, = 0$ and $dist(r_i, s_j)$ equals 0 when $r = s$.

Definition 1. K Nearest Neighbor Join (KNN). For a d-dimensional dataset $S \in \mathfrak{R}^d$ and a query data point r, the KNN operation of r on $\mathbf{S}$ *can be recorded as $knn(r, S, k)$ aiming to obtain the k nearest neighbors of r in* $\mathbf{S}$:

$knn(r, S, k) = \{s_1, s_2, \ldots, s_k | s_i \in S, 1 \le i \le k\}$; for each $\forall s_j \in S - \{s_1, s_2, \ldots, s_k | s_i \in S, 1 \le i \le k\}$, the distance meets the following requirements:

$dist(r, s_1) \le dist(r, s_2) \le \cdots \le dist(r, s_k) \le dist(r, s_j)$.

Definition 2. K Nearest Neighbor Similarity Join (KNN Join). Given two datasets $R \in \mathfrak{R}^d$ and $S \in \mathfrak{R}^d$, the KNN similarity join operation on R and S can return the k nearest neighbors for each data point $r \in$ $\mathbf{R}$ *from S, which can be denoted as $knnJ(\mathbf{RS}) = \{(r, knn(r, S, k)) | \text{for each } r \in \mathbf{R}\}$.*

3.3. Theorems

Theorem 1. *Given two d-dimensional data points $\mathbf{s}_1$ and $\mathbf{s}_2$, then $g(\mathbf{s}_1) - g(\mathbf{s}_2)/dist(\mathbf{s}_1, \mathbf{s}_2) \sim \mathbf{N}(0, 1)$.*

Theorem 2. *Given two d-dimensional data points $\mathbf{s}_1$ and $\mathbf{s}_2$, then $\Delta_m^2(\mathbf{s}_1, \mathbf{s}_2)/dist^2(\mathbf{s}_1, \mathbf{s}_2) \sim \chi^2(m)$.*

Theorem 3. *If $dist(\mathbf{s}_1, \mathbf{s}_2) \le \epsilon$, then the probability that $\Delta_m(\mathbf{s}_1, \mathbf{s}_2)$ is less than or equal to $k\epsilon$ will be bigger than or equal to $1 - P(\chi^2 > k^2)$, which can be denoted as follows: $P(\Delta_m(\mathbf{s}_1, \mathbf{s}_2) \le | k\epsilon \; dist(\mathbf{s}_1, \mathbf{s}_2) \le \epsilon) \ge 1 - P(\chi^2 > k^2)$.*

Theorem 1, Theorem 2, and Theorem 3 have been proved by Ma et al. [64]. Theorem 3 indicates that if the Euclidean distance of the original space is less than or equal to ϵ, the probability that the distance of the projected space will be less than or equal to $k\epsilon$ has the lower bound; that is, $1 - P(\chi^2 > k^2)$. So we can project d-dimensional data point v into m-dimensional space $(m < d)$ through $\pi_m(\mathbf{s}) = \langle g_1(\mathbf{s}), g_2(\mathbf{s}), \ldots, g_m(\mathbf{s})\rangle$.

Theorem 4. *Given three d-dimensional data points $\mathbf{r}, \mathbf{s}_1$, and $\mathbf{s}_2$, $\mathbf{r}, \mathbf{s}_1, \mathbf{s}_2 \in \mathfrak{R}^d$, $\mathbf{U} = \Delta_m^2(\mathbf{r}, \mathbf{s}_1)/dist^2(\mathbf{r}, \mathbf{s}_1) \sim \chi^2(m)$, $\mathbf{V} = \Delta_m^2(\mathbf{r}, \mathbf{s}_2)/dist^2(\mathbf{r}, \mathbf{s}_2) \sim \chi^2(m)$, and then $\mathbf{F} = \mathbf{U}/\mathbf{V}$ obeys the F distribution with degrees of freedom (m,m); that is, $\mathbf{F} = \mathbf{U}/\mathbf{V} \sim F(m, m)$.*

Proof. According to Theorem 2, $\mathbf{U} = \Delta_m^2(\mathbf{r}, \mathbf{s}_1)/dist^2(\mathbf{r}, \mathbf{s}_1) \sim \chi^2(m)$, and $\mathbf{V} = \Delta_m^2(\mathbf{r}, \mathbf{s}_2)/dist^2(\mathbf{r}, \mathbf{s}_2) \sim \chi^2(m)$; that is to say, U and V both obey the χ^2 distribution with freedom m.

Based on the definition of F distribution, $\mathbf{F} = \mathbf{U}/m/\mathbf{V}/m$ obeys the F distribution with degrees of freedom (m,m); that is, $\mathbf{F} = \mathbf{U}/\mathbf{V} \sim F(m, m)$. □ □

Theorem 5. *If $dist(\mathbf{r}, \mathbf{s}_1) \le dist(\mathbf{r}, \mathbf{s}_2)$, then the probability that $\Delta_m(\mathbf{r}, \mathbf{s}_1) \le k\Delta_m(\mathbf{r}, \mathbf{s}_2)$ is bigger than 1-$P(F > k^2)$; that is, $P(\Delta_m(\mathbf{r}, \mathbf{s}_1) \le k\Delta_m(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \le dist(\mathbf{r}, \mathbf{s}_2)) > 1 - P(F > k^2)$. F is the distribution with degrees of freedom (m,m); that is, $F \sim F(m, m)$.*

Proof.

Table 1: Notations.

Notation	Meaning of the notation
n_1, n_2	The data points' number in the dataset.
ϵ	The width of each partition under equal width partition strategy.
d	The data point's dimensionality.
$dist(\mathbf{s}_1, \mathbf{s}_2)$	The Euclidean distance of data point $\mathbf{s}_1$ and data point $\mathbf{s}_2$.
$g(\mathbf{s})$	$g(\mathbf{s}) = \mathbf{a} \cdot \mathbf{s}$, $\mathbf{a}$ is a d-dimensional vector, and each element is a random variable that obeys $\mathbf{p}$-*stable distribution*.
$\pi_m(\mathbf{s})$	$\pi_m(\mathbf{s}) = \langle g_1(\mathbf{s}), g_2(\mathbf{s}), \ldots, g_m(\mathbf{s})\rangle$.
$\Delta_m(\mathbf{s}_1, \mathbf{s}_2)$	$\Delta_m(\mathbf{s}_1, \mathbf{s}_2) = dist(\pi_m(\mathbf{s}_1), \pi_m(\mathbf{s}_2))$.
$\chi^2(m)$	Chi-square distribution with degree of freedom m.
$\pi_1(\mathbf{s})_{\min}$	The minimum projected value of data point s.
$\pi_1(\mathbf{s})_{\max}$	The maximum projected value of data point s.
len	The width of all the projected values in one-dimensional space; that is, $\pi_1(\mathbf{s})_{\min} - \pi_1(\mathbf{s})_{\max}$.
PN	The number of the partitions in one-dimensional space.
P_i	The i_{th} partition.

$$\begin{aligned}
&\because \Delta_m(\mathbf{r}, \mathbf{s}_1) \geq 0 \text{and} \Delta_m(\mathbf{r}, \mathbf{s}_2) \geq 0,\\
&\quad \therefore P(\Delta_m(\mathbf{r}, \mathbf{s}_1) \leq k\Delta_m(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2))\\
&\quad = P\left(\Delta_m^2(\mathbf{r}, \mathbf{s}_1) \leq k^2\Delta_m^2(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)\right)\\
&\quad = P\left(\frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)}{dist^2(\mathbf{r}, \mathbf{s}_1)} \leq \frac{k^2\Delta_m^2(\mathbf{r}, \mathbf{s}_2)}{dist^2(\mathbf{r}, \mathbf{s}_1)} | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)\right)\\
&\quad = \frac{P\left(\Delta_m^2(\mathbf{r}, \mathbf{s}_1)/dist^2(\mathbf{r}, \mathbf{s}_1) \leq k^2\Delta_m^2(\mathbf{r}, \mathbf{s}_2)/dist^2(\mathbf{r}, \mathbf{s}_1) \text{and}\, dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)\right)}{dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)},\\
&\quad \because P(dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)) = 1,\\
&\quad \therefore P(\Delta_m(\mathbf{r}, \mathbf{s}_1) \leq k\Delta_m(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2))\\
&\quad = P\left(\frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)}{dist^2(\mathbf{r}, \mathbf{s}_1)} \leq \frac{k^2\Delta_m^2(\mathbf{r}, \mathbf{s}_2)}{dist^2(\mathbf{r}, \mathbf{s}_1)}\right)\\
&\quad = 1 - P\left(\frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)}{dist^2(\mathbf{r}, \mathbf{s}_1)} > \frac{k^2\Delta_m^2(\mathbf{r}, \mathbf{s}_2)}{dist^2(\mathbf{r}, \mathbf{s}_1)}\right),\\
&\quad \because \frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)}{dist^2(\mathbf{r}, \mathbf{s}_1)} \sim \chi^2(m) \text{and}\, dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2),\\
&\quad \therefore P(\Delta_m(\mathbf{r}, \mathbf{s}_1) \leq k\Delta_m(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2))\\
&\quad > 1 - P\left(\frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)}{dist^2(\mathbf{r}, \mathbf{s}_1)} > \frac{k^2\Delta_m^2(\mathbf{r}, \mathbf{s}_2)}{dist^2(\mathbf{r}, \mathbf{s}_2)}\right)\\
&\quad = 1 - P\left(\frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)/dist^2(\mathbf{r}, \mathbf{s}_1)}{\Delta_m^2(\mathbf{r}, \mathbf{s}_2)/dist^2(\mathbf{r}, \mathbf{s}_2)} > k^2\right),\\
&\quad\quad \text{according to theorem 4,} \frac{\Delta_m^2(\mathbf{r}, \mathbf{s}_1)/dist^2(\mathbf{r}, \mathbf{s}_1)}{\Delta_m^2(\mathbf{r}, \mathbf{s}_2)/dist^2(\mathbf{r}, \mathbf{s}_2)} \sim F(m, m),\\
&\quad \therefore P(\Delta_m(\mathbf{r}, \mathbf{s}_1) \leq k\Delta_m(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{q}, \mathbf{v}_2)) > 1 - P\left(F > k^2\right).
\end{aligned} \tag{2}$$

□

According to Theorem 5, when $k = 1$, if $dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)$, then $P(\Delta_m(\mathbf{r}, \mathbf{s}_1) \leq \Delta_m(\mathbf{r}, \mathbf{s}_2) | dist(\mathbf{r}, \mathbf{s}_1) \leq dist(\mathbf{r}, \mathbf{s}_2)) > 1 - P(F(m, m) > 1)$; it indicates that the probability that $\Delta_m(\mathbf{r}, \mathbf{s}_1) \leq \Delta_m(\mathbf{r}, \mathbf{s}_2)$ has the lower bound: $1 - P(F(m, m) > 1)$.

We can conclude that if $\mathbf{s}_1$ is closer to $\mathbf{r}$ than $\mathbf{s}_2$ in the original d-dimensional space, $\pi_m(\mathbf{s}_1)$ is still likely to be closer to $\pi_m(\mathbf{r})$ than $\pi_m(\mathbf{s}_2)$ in the projected m-dimensional space with lower bound probability $1 - P(F(m, m) > 1)$.

4. Probability Computation

When $m = 1, k = 1$, the probability $P(F(m, m) > 1)$ can be figured out based on the probability density of F distribution which is described as follows:

$$\psi(y) = \begin{cases} \dfrac{\Gamma[(n_1+n_2)/2](n_1/n_2)^{n_1/2} y^{(n_1/2)-1}}{\Gamma(n_1/2)\Gamma(n_2/2)[1+(n_1 y/n_2)]^{(n_1+n_2)/2}}, & y > 0, \\ 0, & \text{other}. \end{cases} \quad (3)$$

Given the freedom (1,1) and the upper quantile with 1, the probability $P(F(m, m) > 1)$ can be calculated as the follows:

$$\begin{aligned} P(F > 1) &= \int_1^\infty \psi(y)dy, \\ &= \int_1^\infty \frac{\Gamma[(1+1)/2](1/1)^{1/2} y^{(1/2)-1}}{\Gamma(1/2)\Gamma(1/2)[1+y]^{(1+1)/2}} dy \\ &= \int_1^\infty \frac{\Gamma(1) y^{-1/2}}{(\Gamma(1/2))^2[1+y]} dy \end{aligned}$$

$$\because \Gamma(1) = 1, \Gamma(1/2) = \sqrt{\pi}$$

$$\begin{aligned} \therefore P(F > 1) &= \int_1^\infty \frac{y^{-1/2}}{(\sqrt{\pi})^2[1+y]} dy \\ &= \frac{1}{\pi}\int_1^\infty \frac{1}{\sqrt{y}(1+y)} dy \\ &= \frac{2}{\pi}\int_1^\infty \frac{1}{\left(1+(\sqrt{y})^2\right)} d\sqrt{y} \\ &= \frac{2}{\pi} arctan\sqrt{y}|_1^\infty \\ &= \frac{2}{\pi}\left(\frac{\pi}{2}-\frac{\pi}{4}\right) = 0.5. \end{aligned} \quad (4)$$

The result of the above computation shows that, given three d-dimensional data points $\mathbf{r}, \mathbf{s}_1$, and $\mathbf{s}_2$, $\mathbf{r}, \mathbf{s}_1, \mathbf{s}_2 \in \mathfrak{R}^d$, they can be reduced to 1-dimensional space through dot product with a d-dimensional vector $\mathbf{a}$. If $\mathbf{s}_1$ is closer to $\mathbf{r}$ than $\mathbf{s}_2$ in the original d-dimensional space, the lower bound probability that $\pi_1(\mathbf{s}_1)$ is still likely to be closer to $\pi_1(\mathbf{r})$ than $\pi_1(\mathbf{s}_2)$ in the projected 1-dimensional space is 0.5.

5. k Nearest Neighbor Join Using Novel Partitioning Strategy

5.1. Algorithm for k Nearest Neighbor Join Using Novel Partitioning Strategy. Theorem 5 shows that if $\mathbf{s}_1$ is closer to $\mathbf{r}$ than $\mathbf{s}_2$ in the original d-dimensional space, $\pi_m(\mathbf{s}_1)$ is still likely to be closer to $\pi_m(\mathbf{r})$ than $\pi_m(\mathbf{s}_2)$ in the projected m-dimensional space with the lower bound probability $1 - P(F(m, m) > 1)$. The conclusion implies that data points in projected space maintain relative location relationship as in original dimensional space. So we proposed k nearest neighbor join algorithm using random projection (RP k NN); it includes two main stages: the first stage is responsible for dimension reduction and space partition, and the second stage is used to conduct k NN join in reduced dimensional space. Figure 1 shows the general framework of k nearest neighbor similarity join algorithm using novel partitioning strategy. The concrete process of the proposed algorithm is described in Algorithm 1. The getPartition routine projects all the data points into one-dimensional space and divides the data points into several partitions according to the specific partition strategy (line 1). For each partition P_i, its corresponding partition $\overline{P_i}$, which needs to be compared with P_i, can be obtained through lines 3–9. If P_i is the leftmost partition, $\overline{P_i} \leftarrow \cup P_i \cup P_{i+1}$ (line 5). If P_i is the rightmost partition, $\overline{P_i} \leftarrow \cup P_{i-1} \cup P_i$ (line 7); in other cases, $\overline{P_i} \leftarrow P_{i-1} \cup P_i \cup P_{i+1}$ (line 9). Finally, for each data point $v \in P_i$, k NN join routine is used to find its k nearest neighbors from $\overline{P_i}$ (lines 10–12).

5.2. Partition Strategy

5.2.1. Equal Width Partition Strategy. All the data points are divided into several partitions with equal width. The total partition number can be set to $PN = \lfloor \sqrt{n} \rfloor$. Suppose that $\pi_1(\mathbf{s})_{\min}$ is the minimum projected value of s in one-dimensional space and $\pi_1(\mathbf{s})_{\max}$ is the maximum projected value of s in one-dimensional space; that is $\pi_1(s)_{\min} = \min\{\pi_1(s_j), s_j \in R\}$ and $\pi_1(s)_{\max} = \max\{\pi_1(s_j), s_j \in R\}$; *len* is the width of all the projected values in one-dimensional space, $len = \pi_1(s)_{\min} - \pi_1(s)_{\max}$; ϵ is the width of each partition, $\epsilon = \lfloor len/PN \rfloor$; given a data point s, its corresponding partition number is $P_i = \lfloor \pi_1(s)/\epsilon \rfloor$. The detailed procedure can be shown in Figure 2 and Algorithm 2.

5.2.2. Distance Split Tree-Based Partition Strategy. The previous equal width partition strategy is easy to implement; however, it cannot deal with skewed dataset efficiently. According to our proposed approach, the d-dimensional data point s will be projected into one-dimensional space through $\pi_1(s) = g_1(s) = v \cdot a = \sum_{i=1}^d s_i * a_i$; each element of a obeys standard normal distribution; that is: $a_i \sim N(0, 1), i \in [1, d]$. The projected value $\pi_1(s)$ is subject to normal distribution, so it is skewed. Figure 3 shows the distribution of the projected values.

Theorem 6. *Given two d-dimensional data points s and a, $a_i \sim N(0, 1), i \in [1, d]$, $\pi_1(s) = g_1(s) = s \cdot a = \sum_{i=1}^d s_i * a_i$, and then $\pi_1(s) \sim N(0, \sum_{i=1}^d s_i^2)$.*

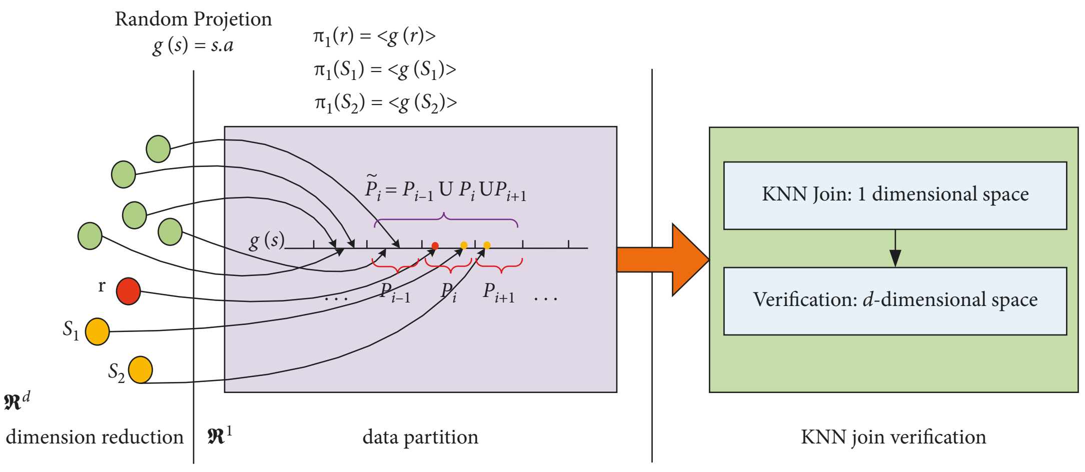


Figure 1: Framework of k nearest neighbor join algorithm using novel partitioning strategy.

```
Input: $R$, $k$ //dataset, the number of nearest neighbors to find
Output: $res$ //a set of pairs of data points
 (1) partitions←get Partition $(R, n)$;
 (2) $res$←$\varnothing$;
 (3) for $i = 1; i \le |\text{partitions}|; i++$ do
 (4) if $i == 1$ then
 (5) $\overline{P_i} \longleftarrow \cup P_i \cup P_{i+1}$
 (6) else if $i ==$ |partitions| then
 (7) $\overline{P_i} \longleftarrow \cup P_{i-1} \cup P_i$
 (8) else
 (9) $\overline{P_i} \longleftarrow P_{i-1} \cup P_i \cup P_{i+1}$
(10) for data point $v \in P_i$ do
(11) temp←$kNN$ Join $(k, v, \overline{P_i})$
(12) $res$←$res \cup$ temp
(13) return res.
```

Algorithm 1: Algorithm for k nearest neighbor join using novel partitioning strategy.

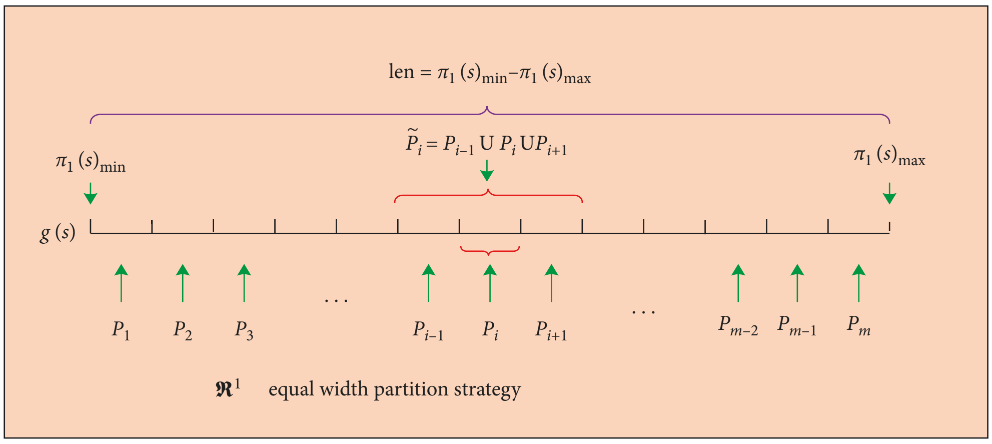


Figure 2: Equal width partition strategy.

```
Input: $R$, $n$ //dataset, the cardinality of the dataset
Output: partitions //list of the sets that partition the dataset
 (1) $PN = \lfloor \sqrt{n} \rfloor$
 (2) partitions$[i] \leftarrow \varnothing, \forall i, 0 \le i \le PN$
 (3) $\pi_1(s)_{\min} = \min\{\pi_1(s_j), s_j \in R\}$
 (4) $\pi_1(s)_{\max} = \max\{\pi_1(s_j), s_j \in R\}$
 (5) $len = \pi_1(s)_{\min} - \pi_1(s)_{\max}$
 (6) $\epsilon = \lfloor len/PN \rfloor$
 (7) choose one vector randomly, recorded as $a, \forall e \in a \sim N(0,1)$
 (8) //divide $\overline{R}$ into $PN$ partitions according to $P_i$, the data points which have the same $P_i$.
 (9) //belong to the same partition, recorded as partitions[1], partitions[2] $\cdots$, partitions[$PN$].
(10) for each data point $s \in R$ do
(11)   $\pi_1(s) \longleftarrow \langle g_1(s) \rangle$
(12)   $P_i = \lfloor \pi_1(s)/\epsilon \rfloor$
(13)   partitions$[P_i] \leftarrow \langle P_i, \pi_1(s), s \rangle$
```

Algorithm 2: Equal width partition strategy.

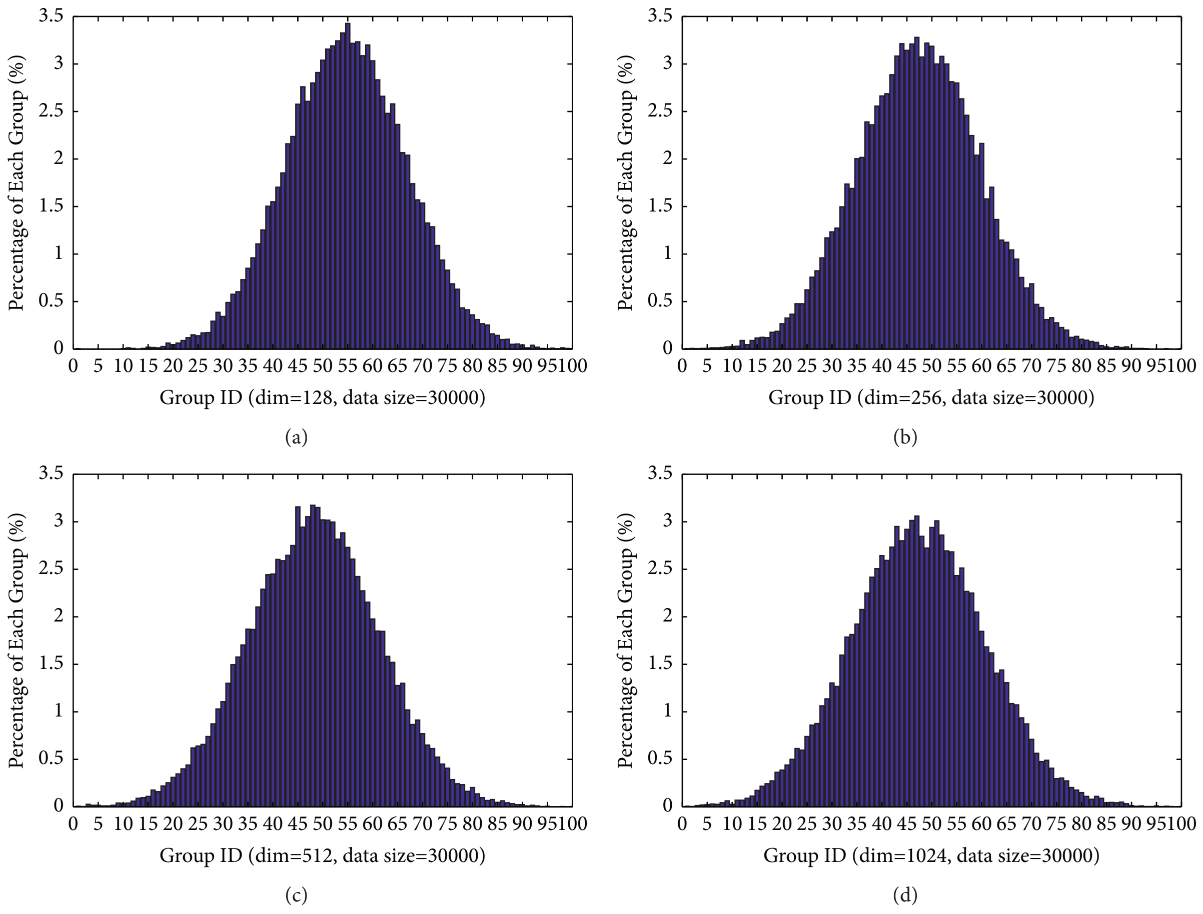


Figure 3: The distribution of the projected values ($\pi_1(s)$): (a) dim = 128 and data science = 30000; (b) dim = 256 and data science = 30000; (c) dim = 512 and data science = 30000; (d) dim = 1024 and data science = 30000.

Proof.

$$\begin{aligned}
&\because a_i \sim N(0,1),\\
&\Rightarrow s_i a_i \sim N\left(0, s_i^2\right),\\
&\because \pi_1(s) = \sum_{i=1}^{d} s_i * a_i,\\
&\therefore \pi_1(s) \sim N\left(0, \sum_{i=1}^{d} s_i^2\right),\\
&\Rightarrow \pi_1(s) \text{is subject to normal distribution and its variance is } \sum_{i=1}^{d} s_i^2.
\end{aligned} \tag{5}$$

Aiming to deal with the data skew problem, we proposed a novel partition strategy called distance split tree- (DST-) based partition strategy. Figure 4 displays the structure of DST. The main idea of DST is that after the original d-dimensional data points are mapped into one-dimensional space, in the beginning, all the data points are divided into equal width partitions with threshold $\epsilon = \pi_1(s)_{\max} - \pi_1(s)_{\min}/2^c$, c is an adjustable parameter. $\max\text{Num}$ is the upper bound of data point count contained in each partition. Once the data point count in a specific partition exceeds $\max\text{Num}$, the partition will be divided into two new partitions with equal width again and so on, and finally, a distance split tree is formed. For each leaf node, the level of the node's hierarchy, the node number in the specific level, the count of the data, and the corresponding dataset are recorded. Based on the above information, the distance range corresponding to each leaf node can be calculated. The corresponding distance width of each node in the current level can be calculated: $1/2^{\text{level}-1}\epsilon$. The corresponding distance range of the node can be calculated by $or\ de\ rNo$ in the level $[\text{order No} - 1/2^{\text{level}-1}\epsilon, \text{order No}/2^{\text{level}-1}\epsilon)$. Thus, the distance range corresponding to the N2 node can be calculated: $[3 - 1/2^{3-1}\epsilon, 3/2^{3-1}\epsilon)$; that is, $[2/4\epsilon, 3/4\epsilon)$.

The construction of distance split tree: the construction process of the distance split tree is as follows: firstly, build a root node N_{root}, for each data point $s_i \in R$, and figure out its projected value in one-dimensional space $\pi_1(s_i)$. Then all the data points are divided into equal width partitions with threshold ϵ, and the corresponding partition number of each vector s_i in the mapping space is obtained $pi\ d \leftarrow \lfloor \pi_1(s_i)/\epsilon \rfloor$. If the node with the number $pi\ d$ does not exist, a new child node with the number $pi\ d$ will be generated. If it already exists, s_i is inserted into the node $pi\ d$, and its count value is increased by 1. Once the amount of data point in a node exceeds a given threshold, such as *maxNum*, the node will be further divided into two subnodes according to the distance range. Repeat this procedure, and finally, a distance split tree is generated.

Data partitions generated: after the distance split tree is constructed, the partitions set can be obtained through preorder traversal for distance split tree; only the leaf nodes are left as the member of the final partitions set. Then the obtained partitions set can be used in Algorithm 1. □

6. Time Complexity Analysis

In this section, we mainly analyze the time complexity of our proposed method. Given the d-dimensional dataset R and $|R| = n$, the total partition number is $PN = \lfloor \sqrt{n} \rfloor$, P_i represents the i_{th} partition, and Cost represents the total computations of the proposed method. The time complexity is as the follows:

$$\text{Cost} = \sum_{i=1}^{PN} \left|P_i\right| * \left|P_i \cup P_{i-1} \cup P_{i+1}\right| * d. \tag{6}$$

In the best cases, all the data points in R are evenly distributed in each partition; that is, $|P_i| = \lfloor \sqrt{n} \rfloor$, so

$$\begin{aligned}
\text{Cost} &= \sum_{i=1}^{PN} \lfloor \sqrt{n} \rfloor * 3 * \lfloor \sqrt{n} \rfloor * d\\
&= \lfloor \sqrt{n} \rfloor * \lfloor \sqrt{n} \rfloor * 3 * \lfloor \sqrt{n} \rfloor * d\\
&= 3 * n^{3/2} * d.
\end{aligned} \tag{7}$$

The time complexity in the best case can be recorded as $\mathcal{O}(n^{3/2}d)$.

In the worst case, supposing that all the data points are included in one partition, then the time complexity should be $\mathcal{O}(n^2 d)$. Because the projected values of the proposed method obey normal distribution, it is between the best case and the worst case, so the time complexity of the proposed method lies in $(\mathcal{O}(n^{3/2}d), \mathcal{O}(n^2 d))$.

7. Experimental Analysis

We conducted experiments to test the effectiveness and performance of the proposed methods, k nearest neighbor join algorithm using random projection with equal width partition strategy (RP k NNEW) and k nearest neighbor join algorithm using random projection with distance split tree-based partition strategy (RP k NNDST), and made comparisons between our proposed methods and the existing

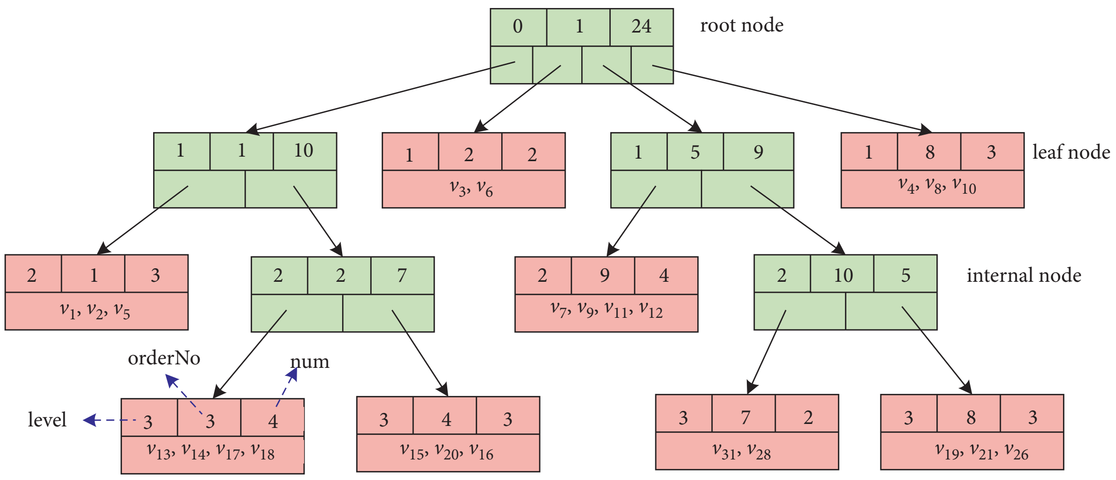


FIGURE 4: Distance split tree-based partition strategy.

methods including *a* KNN [38] and the brute force method with Block Nested Loop Join Strategy (BNLJ). *a* KNN [38] is a relatively new research work on *k* NN similarity join problem, and it also adopted the algorithms based on the partitioning strategy.

Experimental settings: our tests are performed on HP workstation, and the configurations are as follows: CPU: Intel Xeon Gold 6136 @ 3.00 GHz, memory: 128 GB, disk: 2 TB, OS: 64-bit Windows 10, and 12 cores. Table 2 describes the parameters and their values. The bold fonts represent the default values.

Datasets: the datasets adopted in our experiments are synthetic data, the elements of the vector are uniformly distributed in the range [0, 1], and the dimensionality of the datasets includes 128, 256, 512, and 1024. The datasets are listed in Table 3.

7.1. Precision versus Data Size. It can be concluded that RP *k* NNEW has the best precision among the above three approaches, including *a* KNN, RP *k* NNEW, and RP *k* NNDST; in some cases, the precision of RP *k* NNEW is more than 50%. However, Figure 5 shows that the precision of *a* KNN, RP *k* NNEW, and RP *k* NNDST is not very stable for different data size, and the precision of RP *k* NNDST is between that of *a* KNN and RP *k* NNEW.

7.2. Precision versus Dimension. Figure 6 displays the precision of *a* KNN, RP *k* NNEW, and RP *k* NNDST under different dimension. The proposed method RP *k* NNEW has the best precision; although the precision of RP *k* NNEW varies under different dimension, it is always higher than 40% under all conditions. The precision of RP *k* NNDST is relatively stable under different dimension, and it is lower than that of RP *k* NNEW. Because RP *k* NNDST adopts the distance tree-based partition strategy and all the data points will be distributed into different partitions more evenly, every partition will not contain much more data points, so its precision will decrease to some extent compared with RP *k* NNEW. The precision of our proposed methods, including RP *k* NNEW and RP *k* NNDST, is better than that of *a* KNN under all different dimension.

7.3. Precision versus k. Figure 7 displays the precision of the above approaches (including *a* KNN, RP *k* NNEW, and RP *k* NNDST) under different *k*. The results prove that the precision of RP *k* NNEW and RP *k* NNDST is higher than that of the existing method *a* KNN. The precision of RP *k* NNEW is higher than that of RP *k* NNDST; the reason is that RP *k* NNEW adopted the equal width partition strategy; while the projected values are skewed, several partitions will contain more data points; the precision will be higher accordingly. The precision of *a* KNN and RP *k* NNDST is basically stable under different *k* value, while the precision of RP *k* NNEW is a little more sensitive to the different *k* value.

7.4. Precision Distribution. Figure 8 displays the precision distribution of *a* KNN, RP *k* NNEW, and RP *k* NNDST. It can be found that the precision of some data points is very low (less than 5%), and the precision of some data points is very high (more than 80%) by using RP *k* NNEW. The average precision of RP *k* NNEW is higher than that of *a* KNN and RP *k* NNDST; the percentage of the data points whose precision is more than 80% is 20.6%, 0.05%, and 0.57%, respectively, for RP *k* NNEW, *a* KNN, and RP *k* NNDST. The main reason is that RP *k* NNEW adopts the equal width partition strategy, which cannot deal with the skewed projected values effectively; however, RP *k* NNDST adopts distance split tree-based partition strategy, which can distribute all the data points into different partitions more evenly.

7.5. Performance versus Data Size. Figure 9 displays the performance of BNLJ, *a* KNN, RP *k* NNEW, and RP *k* NNDST on the datasets with different size. The run time of *a* KNN, RP *k* NNEW, and RP *k* NNDST is much less than that of the BNLJ method; the run time of BNLJ increases exponentially with the size of the datasets; however, the run time of *a* KNN, RP *k* NNEW, and RP *k* NNDST increases

Table 2: Experimental settings.

Experimental parameters	Values of the parameters
Returned number: k	10, 20, 30, 40, and 50
Dimensionality: d	128, 256, 512, and 1024
Data size: N	10000, 20000, 30000, 40000, and 50000

Table 3: Datasets descriptions.

Dataset	Number	Dim.	Data size (M)
Data-128-1	10,000	128	11.3
Data-128-2	20,000	128	22.5
Data-128-3	30,000	128	33.8
Data-128-4	40,000	128	45
Data-128-5	50,000	128	56.3
Data-256-3	30,000	256	67.6
Data-512-1	10,000	512	45
Data-512-2	20,000	512	90
Data-512-3	30,000	512	135.1
Data-512-4	40,000	512	180.1
Data-512-5	50,000	512	225.1
Data-1024-3	30,000	960	270

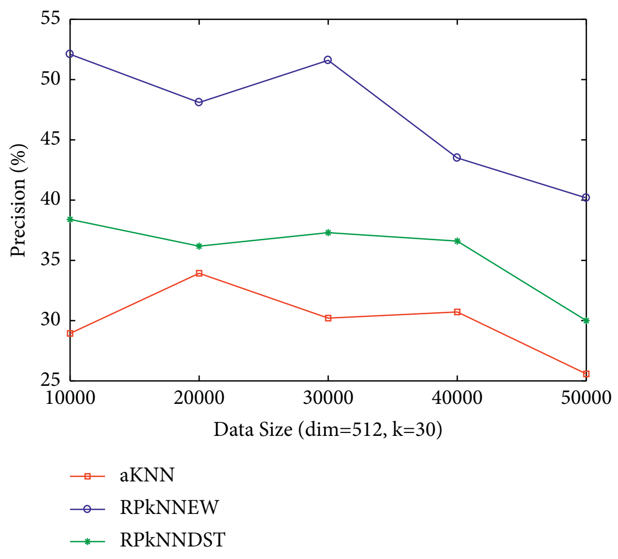


Figure 5: Precision versus data size.

linearly with the size of the datasets. The run time of RP k NNEW is a little bit more than that of a KNN and RP k NNDST. While we can find that RP k NNEW has the best precision among all the methods based on the above precision analysis, we can choose the RP k NNEW method when the performance requirements are not very strict; otherwise, RP k NNDST will be the most appropriate choice, because the precision of RP k NNDST is higher than that of a KNN, while its run time is less than that of RP k NNEW.

7.6. Performance versus Dimension. Figure 10 displays the performance of our proposed methods and the existing methods for different dimensions, which are 128, 256, 512, and 1024, respectively. The time of all algorithms grows with the increase of the dimension, and the reason is the bigger the dimension, the higher the time complexity. The performance of RP k NNEW is the best when the dimension is less than 512, while the run time of RP k NNEW will be slightly higher than that of a KNN when the dimension exceeds 512. The run time of RP k NNEW is higher than that of a KNN and RP k NNEW, while Figure 6 shows that RP k NNEW has the best precision in all cases.

7.7. Performance versus k. The performance of BNLJ, a KNN, RP k NNEW, and RP k NNDST with different k (data size = 30000; dim = 512) is displayed in Figure 11. The run

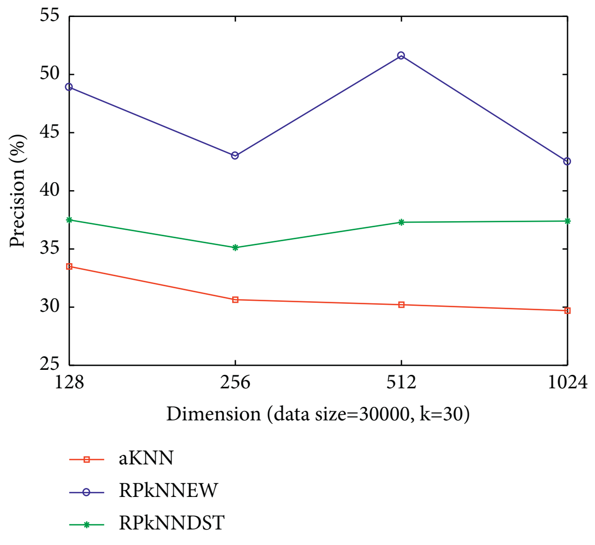


Figure 6: Precision versus dimension.

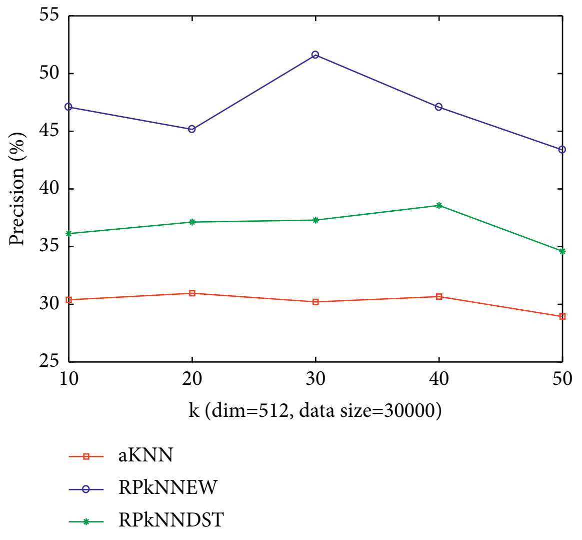


Figure 7: Precision versus *k*.

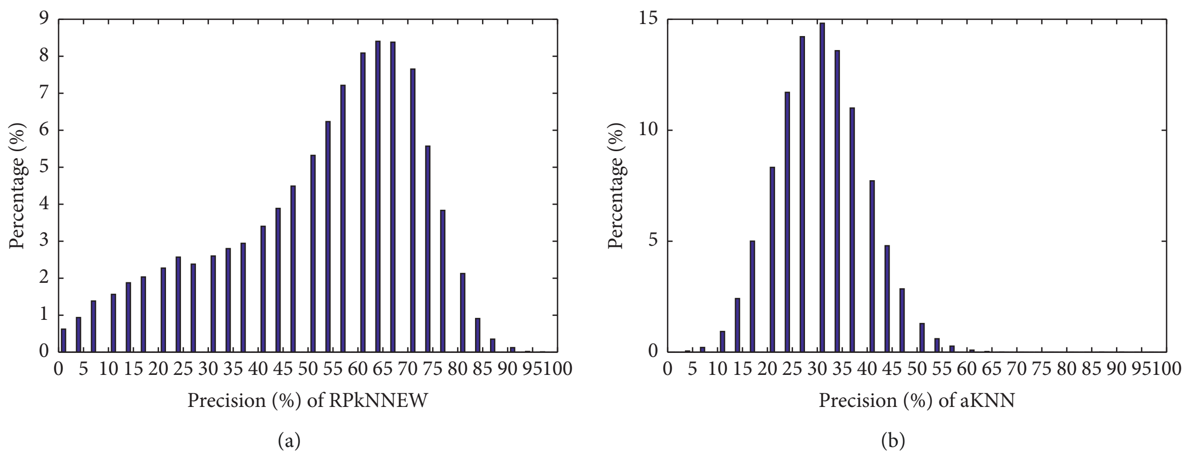
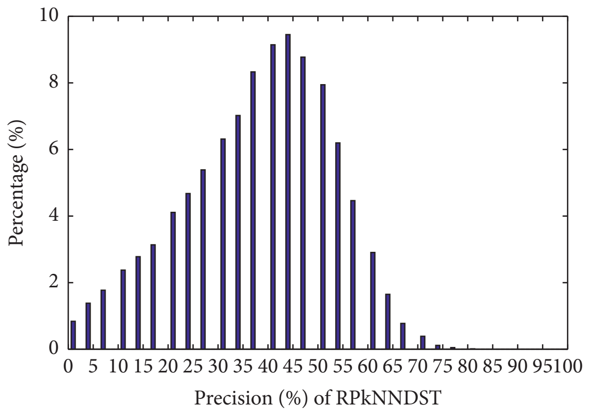
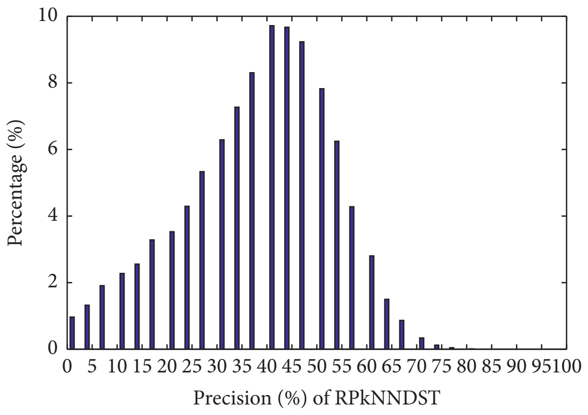


Figure 8: Continued.



(c)



(d)

Figure 8: The distribution of the precision: (a) dim = 512, data size = 30000, and k = 30; (b) dim = 512, data size = 30000, and k = 30; (c) dim = 512, data size = 30000, and k = 30; (d) dim = 1024, data size = 30000, and k = 30.

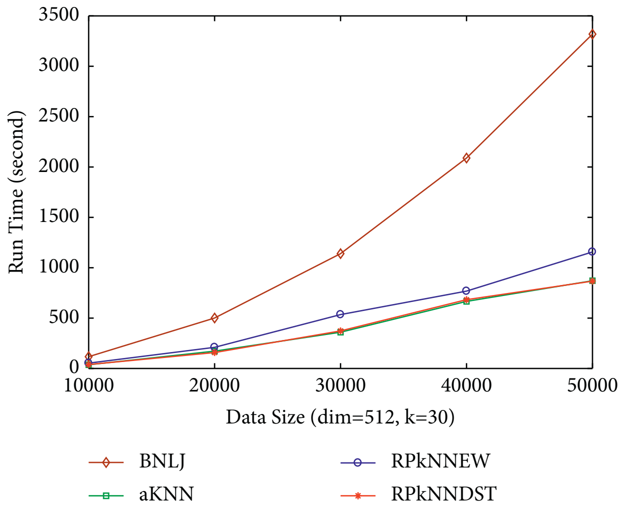


Figure 9: Performance versus data size.

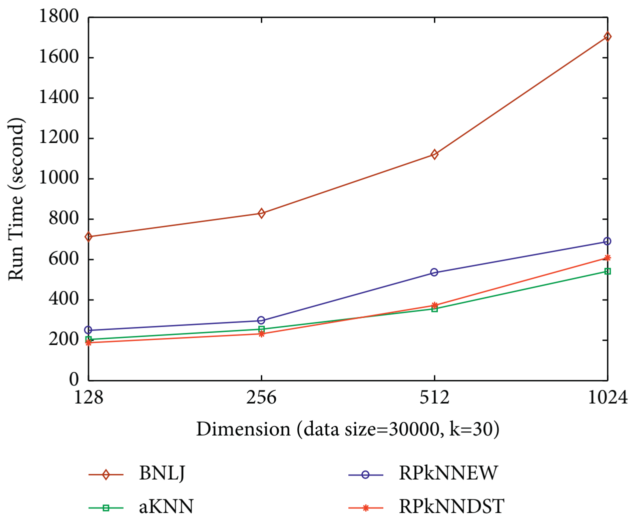


Figure 10: Performance versus dimension.

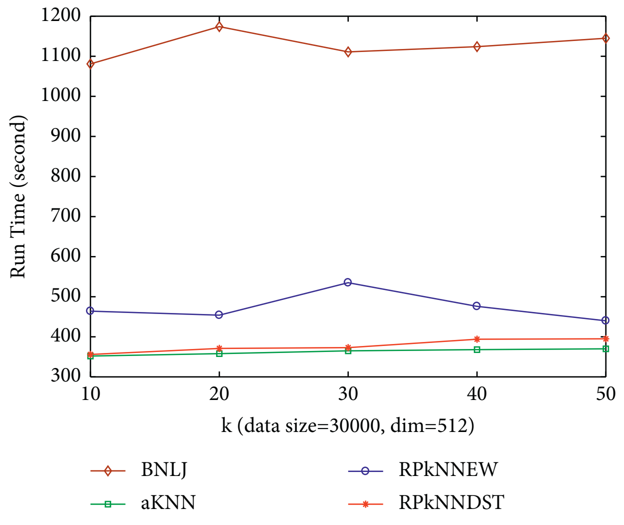


Figure 11: Performance versus k.

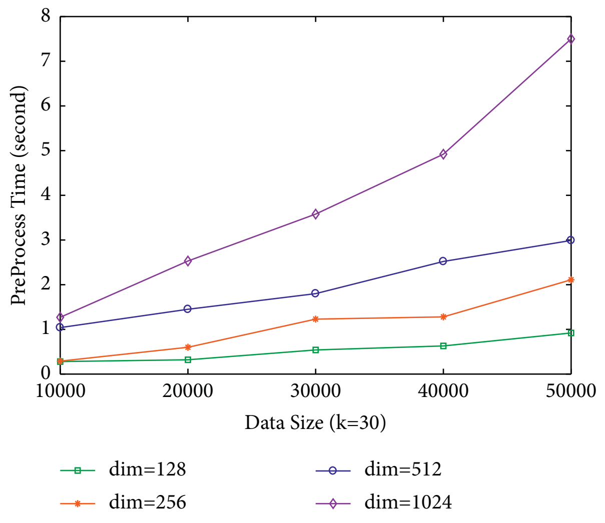


Figure 12: Preprocessing time.

time of all the above algorithms changes little with the different value of k. The run time of RP k NNDST is very close to that of a KNN, and the reason has been explained in Section 6; however, the precision of RP k NNDST is better than that of a KNN according to Figure 7. The run time of RP k NNDST and a KNN is less than that of RP k NNEW.

7.8. Preprocessing Time. Figure 12 displays the preprocessing time required by the construction of the distance split tree (DST) in RP k NNDST approach for different dataset. The preprocessing time increases almost linearly with the size of the dataset. Given a dataset with a fixed size, the preprocessing time increases exponentially with the growth of the dimension. Overall, the proportion of preprocessing time in the total time is low and relative stable. The benefit of the distance split tree (DST) can make up for the additional overhead caused by the construction of DST and can make the total run time of RP k NNDST less than that of RP k NNEW.

8. Conclusions

In the above sections, we mainly studied the k nearest neighbor similarity join problem on high-dimensional data. We proposed k nearest neighbor join algorithm using random projection with equal width partition strategy (RP k NNEW) and k nearest neighbor join algorithm using random projection with distance split tree-based partition strategy (RP k NNDST), which can filter out many unnecessary comparisons and ensure the required precision. We also conducted several experiments to test the effectiveness and performance of our proposed approaches, and the test results show that the proposed approaches in this paper have better effectiveness and performance. However, the proposed approaches in this paper have some limitations, and they can only work with the Euclidean distance. In future research works, we are planning to further study the k NN similarity join approaches, which can deal with other similarity measures, other more effective dimension reduction techniques, and the distributed k nearest neighbor similarity join algorithms.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by the grants from the Science and Technology Research Plan Project of Henan Province (202102210357); the Innovative Research Team (in Science and Technology) at University of Henan Province (22IRTSTHN016); the National Natural Science Foundation of China (61602231); the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) (JP21K17737); and the Hubei Natural Science Foundation (2021CFB156).

References

[1] W. L. Shang, J. Y. Chen, H. B. Bi, Y. Sui, Y. Chen, and H. Yu, "Impacts of covid-19 pandemic on user behaviors and environmental benefits of bike sharing: a big-data analysis," *Applied Energy*, vol. 285, 2021.

[2] G. T. Reddy, M. P. K. Reddy, and K. Lakshmanna, "Analysis of dimensionality reduction techniques on big data," *IEEE Access*, vol. 8, Article ID 54776, 2020.

[3] L. Zhen, Y. K. Zhang, K. P. Yu, N. Kumar, and A. Barnawi, "Early collision detection for massive random access in satellite-based internet of things," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 5184–5189, 2021.

[4] Z. W. Guo, K. P. Yu, A. Jolfaei, F. Ding, and N. Zhang, "Fuzspam: label smoothing-based fuzzy detection of spammers in internet of things," *IEEE Transactions on Fuzzy Systems*, 2021.

[5] K. P. Yu, L. Tan, L. Lin, X. Cheng, Z. Yi, and T. Sato, "Deep-learning-empowered breast cancer auxiliary diagnosis for 5gb remote e-health," *IEEE Wireless Communications*, vol. 28, no. 3, pp. 54–61, 2021.

[6] F. Ding, G. P. Zhu, M. Alazab, X. J. Li, and K. P. Yu, "Deep-learning-empowered digital forensics for edge consumer electronics in 5g hetnets," *IEEE Consumer Electronics Magazine*, vol. 11, 2020.

[7] F. Ding, G. P. Zhu, Y. C. Li, X. Zhang, P. K. Atrey, and S. Lyu, "Anti-forensics for face swapping videos via adversarial training," *IEEE Transactions on Multimedia*, 2021.

[8] F. Ding, K. P. Yu, Z. H. Gu, X. J. Li, and Y. Q. Shi, "Perceptual enhancement for autonomous vehicles: restoring visually degraded images for context prediction via adversarial training," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[9] W. Z. Wang, M. H. Fida, Z. T. Lian et al., "Secure-enhanced federated learning for ai-empowered electric vehicle energy prediction," *IEEE Consumer Electronics Magazine*, 2021.

[10] L. Tan, K. P. Yu, L. Lin, X. Cheng, G. Srivastava, and W. Wei, "Speech emotion recognition enhanced traffic efficiency solution for autonomous vehicles in a 5g-enabled space-air-ground integrated intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, 2021.

[11] C. S. Feng, B. Liu, K. P. Yu, S. K. Goudos, and S. H. Wan, "Blockchain-empowered decentralized horizontal federated learning for 5g-enabled uavs," *IEEE Transactions on Industrial Informatics*, vol. 18, 2021.

[12] C. S. Feng, B. Liu, Z. Guo, K. Yu, and Z. Qin, "Blockchain-based cross-domain authentication for intelligent 5g-enabled internet of drones," *IEEE Internet of Things Journal*, vol. 9, no. 8, 2021.

[13] L. Yang, K. P. Yu, S. X. Y. Yang, C. Chakraborty, and Y. Lu, "An intelligent trust cloud management method for secure clustering in 5g enabled internet of medical things," *IEEE Transactions on Industrial Informatics*, 2021.

[14] D. W. Wang, Y. X. He, K. P. Yu, L. Nie, and R. Zhang, "Delay sensitive secure noma transmission for hierarchical hap-lap medical-care iot networks," *IEEE Transactions on Industrial Informatics*, 2021.

[15] H. Li, K. P. Yu, B. Liu, C. Feng, and Z. Qin, "An efficient ciphertext-policy weighted attribute-based encryption for the internet of health things," *IEEE journal of biomedical and health informatics*, 2021.

[16] Y. Sun, J. Liu, K. P. Yu, M. Alazab, and K. X. Lin, "Pmrss: privacy-preserving medical record searching scheme for intelligent diagnosis in iot healthcare," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1981–1990, 2022.

[17] L. Tan, K. P. Yu, N. Shi, C. Yang, W. Wei, and H. Lu, "Towards secure and privacy-preserving data sharing for covid-19 medical records: a blockchain-empowered approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 271–281, 2022.

[18] H. Xiong, C. J. Jin, M. Alazab, H. Wang, W. Wang, and C. Su, "On the design of blockchain-based ecdsa with fault-tolerant batch verication protocol for blockchain-enabled iomt," *IEEE journal of biomedical and health informatics*, 2021.

[19] W. Z. Wang, C. Qiu, Z. M. Yin, G. Srivastava, and C. Su, "Blockchain and puf-based lightweight authentication protocol for wireless medical sensor networks," *IEEE Internet of Things Journal*, 2021.

[20] C. S. Feng, K. P. Yu, M. Aloqaily, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent iov," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, Article ID 13784, 2020.

[21] Q. Zhang, K. P. Yu, Z. W. Guo, S. Garg, J. Rodrigues, and M. Guizani, "Graph neural networks-driven traffic forecasting for connected internet of vehicles," *IEEE Transactions on Network Science and Engineering*, 2021.

[22] K. P. Yu, L. Tan, C. X. Yang, A. K. Bashir, and T. Sato, "A blockchain-based shamir's threshold cryptography scheme for data protection in industrial internet of things settings," *IEEE Internet of Things Journal*, 2021.

[23] D. Y. Xu, K. P. Yu, and J. A. Ritcey, "Cross-layer device authentication with quantum encryption for 5g enabled iiot in industry 4.0," *IEEE Transactions on Industrial Informatics*, 2021.

[24] K. P. Yu, L. Tan, S. Mumtaz, S. A. Rubaye, A. A. Dulaimi, and A. K. Bashir, "Securing critical infrastructures: deep-learning-based threat detection in iiot," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 76–82, 2021.

[25] Y. Gong, L. Zhang, R. P. Liu, K. P. Yu, and G. Srivastava, "Nonlinear mimo for industrial internet of things in cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5533–5541, 2021.

[26] L. Tan, K. P. Yu, F. P. Ming, X. F. Chen, and G. Srivastava, "Secure and resilient artificial intelligence of things: a honeynet approach for threat detection and situational awareness," *IEEE Consumer Electronics Magazine*, 2021.

[27] K. P. Yu, Z. W. Guo, Y. Shen, W. Wang, and T. Sato, "Secure artificial intelligence of things for implicit group recommendations," *IEEE Internet of Things Journal*, vol. 9, 2021.

[28] T. Guo, K. P. Yu, M. Aloqaily, and S. H. Wan, "Constructing a prior-dependent graph for data clustering and dimension reduction in the edge of aiot," *Future Generation Computer Systems*, vol. 128, pp. 381–394, 2022.

[29] Y. H. Peng, A. Jolfaei, and K. P. Yu, "A novel real-time deterministic scheduling mechanism in industrial cyber-physical systems for energy internet," *IEEE Transactions on Industrial Informatics*, 2021.

[30] L. Zhao, H. Chai, Y. Han, K. Yu, and S. Mumtaz, "A collaborative v2x data correction method for road safety," *IEEE Transactions on Reliability*, 2022.

[31] Z. Zhou, X. Dong, Z. Li, K. Yu, C. Ding, and Y. Yang, "Spatio-temporal feature encoding for traffic accident detection in vanet environment," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[32] L. Cai, J. Y. Gao, and D. Zhao, "A review of the application of deep learning in medical image classification and segmentation," *Annals of Translational Medicine*, vol. 8, no. 11, 2020.

[33] J. X. Zhuang, J. B. Cai, R. X. Wang, J. G. Zhang, and W. S. Zheng, "Deep knn for medical image classification," in *Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, pp. 127–136, Springer International Publishing, Berlin, Germany, 2020.

[34] H. A. Li, M. Zhang, Z. H. Yu, Z. L. Li, and N. Li, "An improved pix2pix model based on gabor filter for robust color image rendering," *Mathematical Biosciences and Engineering*, vol. 19, no. 1, pp. 86–101, 2021.

[35] H. A. Li, Q. Y. Zheng, W. J. Yan, and R. Tao, "Image super-resolution reconstruction for secure data transmission in internet of things environment," *Mathematical Biosciences and Engineering*, vol. 18, no. 5, pp. 6652–6671, 2021.

[36] J. Pang, Y. Gu, J. Xu, and G. Yu, "Research advance on similarity join queries," *Journal of Frontiers of Computer Science and Technology*, vol. 7, no. 1, pp. 1–13, 2013.

[37] Y. Z. Ma, Z. H. Zhang, and C. J. Lin, "Research progress in similarity join query of big data," *Journal of Computer Applications*, vol. 38, no. 4, pp. 978–986, 2018.

[38] S. Ferrada, B. Bustos, and N. Reyes, "An efficient algorithm for approximated self-similarity joins in metric spaces," *Information Systems*, vol. 91, 2020.

[39] W. Lu, Y. Y. Shen, S. Chen, and C. B. Ooi, "Efficient processing of k nearest neighbor joins using mapreduce," *Proceedings of the VLDB Endowment*, vol. 5, no. 10, pp. 1016–1027, 2012.

[40] J. Dai and Z. M. Ding, "Mapreduce based fast knn join," *Chinese Journal of Computers*, vol. 38, no. 1, pp. 99–108, 2015.

[41] X. J. Zhao, J. F. Zhang, and X. Qin, "knn-dp: handling data skewness in knn joins using mapreduce," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 600–613, 2018.

[42] G. Song, J. Rochas, and F. Huet, "Solutions for processing k nearest neighbor joins for massive data on mapreduce," in *Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp. 279–287, Turku, Finland, March 2015.

[43] G. Song, J. Rochas, E. L. Beze, and F. Huet, "K nearest neighbour joins for big data on mapreduce: a theoretical and experimental analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2376–2392, 2016.

[44] A. Stupar, S. Michel, and R. Schenkel, *Rankreduce-processing K-Nearest Neighbor Queries on Top of Mapreduce*, LSDS-IR@ SIGIR, 2010.

[45] Y. Hu, G. Peng, Z. H. Wang, Y. R. Cui, and H. Qin, "Partition selection for large-scale data management using knn join processing," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–14, Article ID 7898230, 2020.

[46] Y. X. Fang, R. Cheng, W. B. Tang, S. Maniu, and S. X. Yang, "Scalable algorithms for nearest-neighbor joins on big trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 785–800, 2016.

[47] G. Chatzigeorgakidis, S. Karagiorgou, S. Athanasiou, and S. Skiadopoulos, "Fml-knn: scalable machine learning on big data using k-nearest neighbor joins," *Journal of Big Data*, vol. 5, no. 1, 2018.

[48] Q. S. Du and X. F. Li, "A novel knn join algorithms based on hilbert r-tree in mapreduce," in *Proceedings of the 3rd International Conference on Computer Science and Network Technology*, pp. 417–420, Dalian, China, October 2013.

[49] Y. Kim and K. Shim, "Parallel top-k similarity join algorithms using mapreduce," in *Proceedings of the IEEE 28th International Conference on Data Engineering*, pp. 510–521, Arlington, VA, USA, 2012.

[50] D. H. Chen, C. G. Shen, J. Y. Feng, and J. J. Le, "An efficient parallel top-k similarity join for massive multidimensional data using spark," *International journal of database theory and application*, vol. 8, no. 3, pp. 57–68, 2015.

[51] Y. Z. Ma, X. Ci, and X. F. Meng, "Parallel top-k join on massive high-dimensional vectors," *Chinese Journal of Computers*, vol. 38, no. 1, pp. 86–98, 2015.

[52] B. Lei, J. Xu, Y. Gu, and G. Yu, "Parallel top-k similarity join algorithm on probabilistic data based on earth mover's distance," *Journal of Software*, vol. 24, no. s2, pp. 188–199, 2013.

[53] J. Huang, R. Zhang, R. Buyya, and J. Chen, "Melody-join: efficient earth mover's distance similarity joins using mapreduce," in *Proceedings of the IEEE 30th International Conference on Data Engineering*, pp. 808–819, Chicago, IL, USA, March 2014.

[54] X. Jia, B. Lei, Y. Gu, and Z. Zhang, "Efficient similarity join based on earth mover's distance using mapreduce," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2148–2162, 2015.

[55] J. Huang, R. Zhang, R. Buyya, J. Chen, and Y. W. Wu, "Heads-join: efficient earth mover's distance similarity joins on hadoop," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1660–1673, 2016.

[56] T. Christiani, R. Pagh, and J. Sivertsen, "Scalable and robust set similarity join," in *Proceedings of the IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 1240–1243, Paris, France, April 2018.

[57] M. Gowanlock and B. Karsin, "Accelerating the similarity self-join using the gpu," *Journal of Parallel and Distributed Computing*, vol. 133, no. 9, pp. 107–123, 2019.

[58] L. N. Yu, T. Z. Nie, D. R. Shen, and Y. Kou, "An approach for progressive set similarity join with gpu accelerating," in *Proceedings of the Web Information Systems and Applications. WISA*, pp. 155–167, 2020.

[59] F. Shao, G. Chen, L. H. Yu, Y. J. Bei, and J. X. Dong, "Bitmap filtering: an efficient speedup method for xml structural matching," in *Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence,*

Networking, and Parallel/Distributed Computing (SNPD 2007), vol. 3, pp. 756–761, Qingdao, China, July 2007.

[60] X. F. Ding, W. L. Yang, R. K. K. Choo, X. L. Wang, and H. Jin, "Privacy preserving similarity joins using mapreduce," *Information Sciences*, vol. 493, pp. 20–33, 2019.

[61] J. C. Wu, Y. Zhang, J. Wang, C. Lin, Y. Fn, and C. Xing, "Scalable metric similarity join using mapreduce," in *Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1662–1665, Macao, China, April 2019.

[62] Y. Z. Ma, X. F. Meng, and S. Y. Wang, "Parallel similarity joins on massive high dimensional data using mapreduce," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 1, pp. 166–183, 2016.

[63] Y. Z. Ma, S. J. Jia, and Y. X. Zhang, "A novel approach for high dimensional vector similarity join query," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 5, 2017.

[64] Y. Z. Ma, S. J. Jia, and Y. X. Zhang, "Chi-square distribution based similarity join query algorithm on high-dimensional data," *Journal of Computer Applications*, vol. 36, no. 7, pp. 1993–1997, 2016.